CPSC 467b: Cryptography and Computer Security

Michael J. Fischer

Lecture 2
January 13, 2010

Recall

A *symmetric cryptosystem* consists of

- a set $\mathcal{M}$ of *plaintext messages*,
- a set $\mathcal{C}$ of *ciphertexts*,
- a set $\mathcal{K}$ of keys,
- an *encryption* function $E : \mathcal{K} \times \mathcal{M} \to \mathcal{C}$
- a *decryption* function $D : \mathcal{K} \times \mathcal{C} \to \mathcal{M}$.

We often write $E_k(m) = E(k, m)$ and $D_k(c) = D(k, c)$.

Decipherability $\forall m \in \mathcal{M}, \forall k \in \mathcal{K}, D_k(E_k(m)) = m$. In other words, D_k is the left inverse of E_k.

Feasibility E and D, regarded as functions of two arguments, should be computable using a feasible amount of time and storage.

Security (weak) It should be difficult to find m given $c = E_k(m)$ without knowing k.

What's wrong with this definition?

This definition leaves several important questions unanswered?

1. What is a "feasible" amount of time and storage?
   This is the work that Alice and Bob must do.
2. What information are Alice and Bob really trying to protect?
   What would they regard a successful attack by Eve?
   Suppose Eve could discover the first bit of m? Is that bad?
3. What assumptions can they reasonably make about Eve?
   How much computing power does she have at her disposal?
   How much information has she acquired in the past that might help her decipher c?
4. What randomness assumptions are we willing to make?
   How are the messages chosen?
   How are the keys chosen?
   What sources of randomness are available to Eve?

Modern cryptography

The goal of *Modern Cryptography* is to make mathematically precise definitions of security so that the security of cryptographic primitives can be proven to hold.

We will follow Katz and Lindell long enough to give a flavor of how definitions can be made precise and to learn what it takes to actually prove security properties.

We now discuss each of the four questions in some depth.

Measuring computational difficulty

We want a notion of how much time is required to compute the encryption and decryption functions.

Why not use actual running time?

- It depends on the speed of the computer as well as on the algorithm for computing the function.
- It varies from one input to another.
- It is difficult to analyze at a fine grained level of detail.

Role of complexity theory

Complexity theory allows one to make meaningful statements about the difficulty of computational problems, independent of the particular computer or programming language.

Complexity measures *rate of growth* of worst-case running time as a function of the length n of the inputs.

An algorithm runs in time $T(n)$ if its running time on all but finitely many inputs x is at most $T(|x|)$.

An algorithm A runs in *polynomial time* if it runs in time $p(n)$ for some polynomial function $p(n)$.

A function f is *polynomial time* if it is computable by some polynomial time algorithm.

Feasibility

Polynomial time functions are said to be *feasible*.

Feasibility is a minimal requirement.

In practice, E and D need to be computable fast enough so that the computation cost does not overwhelm the value of the security gains.

Confidential information

A cryptosystem is compromised if an attacker obtains confidential information.

What is considered confidential depends on the application.

In our simple example, we assumed an all-or-nothing notion of security—either Eve obtains m or she does not.

Naively, we'd like to require that Eve can *never* obtain m.
Even then, we must be careful about what it means to "obtain" m.
Suppose Eve simply guesses m without looking at c.
She will be right with probability $1/|\mathcal{M}|$.
Is this a successful attack?

A more nuanced approach

Compromises of decreasing difficulty for Eve.

Complete break Eve can find the key k.

Can read all communications between Alice and Bob.
Can impersonate Alice when talking to Bob.
Can impersonate Bob when talking to Alice.

Complete message recovery Eve can decrypt arbitrary messages m.

Can read all communications between Alice and Bob.
Cannot impersonate Alice or Bob to the other.

Selected message recovery Eve can decrypt some subset $M \subseteq \mathcal{M}$ of messages.

The larger M is, the more serious the compromise.

Attacks that do not always succeed

Eve doesn't always have to succeed to do damage.

Uncertain message recovery Eve can decrypt messages encrypted with keys from some subset $K \subseteq \mathcal{K}$ of "weak" keys. The larger K is, the more serious the compromise. Eve's probability of success is $|K|/|\mathcal{K}|$ (if k random).

Probabilistic algorithms Eve may use a probabilistic algorithm that only succeeds with some probability > 0.

Summary: Whether or not Eve succeeds in compromising the system depends on many variables.
We want that she succeeds with only a "negligible" probability.

Partial information

Even partial information about m may be considered confidential, so the following might also compromise the system.

Partial message recovery Eve obtains partial information about m.

Initially, Eve knows message probability distribution.
After seeing $c = E_k(m)$, she may learn that m lies in a proper subset $M \subseteq \mathcal{M}$.
If $|M| = 1$, she learns m. If $|M| \geq 2$, she doesn't *know* m for certain, but she knows a lot about m.

As before, the threat might be credible even if Eve only succeeds on some messages or some keys or with some probability on vulnerable messages and keys.

Eve's information

Until now, we've implicitly assumed that Eve knows nothing about the cryptosystem except for the ciphertext c.

In practice, Eve might know much more.

- She probably knows (or has a good idea) of the message distribution.
- She might have obtained several other ciphertexts.
- She might have learned the decryptions of earlier ciphertexts.
- She might have even chosen the earlier messages or ciphertexts herself.

This leads us to consider several attack scenarios.

Attack scenarios

Ciphertext-only attack Eve knows only c and tries to recover m.

Known plaintext attack Eve knows c and a sequence of plaintext-ciphertext pairs $(m_1, c_1), \ldots, (m_r, c_r)$ where $c \notin \{c_1, \ldots, c_r\}$. She tries to recover m.

Chosen plaintext attack Like known plaintext attack, except that she chooses messages $m_1, \ldots, m_r$ before getting c and gets Alice (or Bob) to encrypt them for her.

Chosen ciphertext attack Like a known plaintext attack, except that she chooses ciphertexts $c_1, \ldots, c_r$ before getting c and gets Alice (or Bob) to decrypt them for her.

Still stronger attack scenarios

In an *adaptive chosen plaintext attack*, Eve chooses the messages m_i one at a time rather than all at once. Thus, m_2 depends on c_1, m_3 depends on c_1 and c_2, etc.

Adaptive chosen ciphertext attacks are similarly possible.

In a *mixed chosen plaintext/chosen ciphertext attack*, Eve chooses some plaintexts and some ciphertexts and gets the corresponding decryptions or encryptions.

Adaptive versions of mixed attacks are similarly possible.

Can they really happen?

A known plaintext attack occurs when Alice uses the same key to encrypt several messages, and Eve intercepts the ciphertexts.

Why would Alice cooperate in a chosen plaintext attack?

- "Alice" might be an Internet server, not a person, who encrypts/decrypts certain messages received in the course of carrying out a more complicated cryptographic protocol. (We will see such protocols later in the course.)
- Eve might sit down at Alice's computer when Alice is away.
- Eve might slip Alice's cryptographic smart card out of her purse.
- Eve might be authorized to generate messages that are then encrypted and sent to Bob, but she isn't authorized to read other people's messages.

Randomness in cryptography

Where do we assume randomness?

1. The message is drawn at random from some arbitrary probability distribution over the message space $\mathcal{M}$ which is part of Eve's *a priori* knowledge.
2. The secret key is chosen uniformly at random from the key space $\mathcal{K}$.
3. Eve has access to a source of randomness which she may use while attempting to break the system. For example, Eve can choose an element $k' \in \mathcal{K}$ at random. With probability $p = 1/|\mathcal{K}|$, her element k' is actually the correct key k.

Independence

The three sources of randomness are assumed to be *statistically independent*.

Eve's random numbers do not depend on (nor give any information about) the message or key used by Alice.

Alice's key does not depend on the particular message or vice versa.

Joint probability distribution

These multiple sources of randomness give rise to a *joint probability distribution* that assigns a well-defined probability to each triple (m, k, z), where m is a message, k a key, and z is the result of the random choices Eve makes during her computation.

The independence assumption implies that

$$P[m, k, z] = P[m] \times P[k] \times P[z]$$

where

- $P[m]$ is the probability that m is the chosen message
- $P[k]$ is the probability that k is the chosen key
- $P[z]$ is the probability that z represents Eve's random choices.

Eve's success probability

The joint distribution gives rise to an overall success probability for Eve (once we decide on what it means for an attack to succeed).

We want Eve's success probability to be "small".

Here, "small" is measured relative to a *security parameter*, which you can think of as the key length.

Definition

A function f is *negligible* if for every polynomial $p(\cdot)$ there exists an N such that for all integers $n > N$ it holds that $f(n) < \frac{1}{p(n)}$.

We require that the success probability be a negligible function of the security parameter.

Computational security

Putting this all together, we get a general notion of computational security.

Definition

A cryptosystem is *computationally secure* relative to a notion of compromise if, for all probabilistic polynomial-time algorithms $\mathcal{A}$, when given as input the security parameter n and all of the information available to Eve, the algorithm succeeds in compromising the cryptosystem with success probability that is negligible in n.

Katz and Lindell discuss computational security in greater depth in chapter 3.

Practical security considerations

In practice, the important tradeoff is between the amount of time that Alice and Bob are willing to spend to use the cryptosystem versus what a determined adversary might be willing to spend to break the system.

Asymptotic complexity results will not tell us how to set the security parameter for a system, but they may inform us about how much security improvement we can expect as the key length increases.

Probability distributions and events

We give a quick overview of probability theory.

A *discrete probability distribution* p assigns a real number $p_\omega \in [0, 1]$ to each element ω of a probability space Ω such that

$$\sum_{\omega \in \Omega} p_\omega = 1.$$

An *event* E is a subset of Ω. The probability of E is

$$P[E] = \sum_{\omega \in E} p_\omega.$$

Random variables

A *random variable* is a function $X : \Omega \to \mathcal{X}$, where $\mathcal{X}$ is a set.

We think of X as describing a random choice according to distribution p.

Let $x \in \mathcal{X}$. Event $X = x$ means that the outcome of choice X is x.

Formally, the event $X = x$ is the set $\{\omega \in \Omega \mid X(\omega) = x\}$.
Its probability is therefore

$$P[x = X] = \sum_{\omega : X(\omega) = x} p_\omega .$$

We sometimes ambiguously write x to denote the event $X = x$.

Experiments

Sometimes m denotes the random variable that describes the experiment of Alice choosing a message $m \in \mathcal{M}$ according to the assumed message distribution.

Other times, m denotes a particular message in set $\mathcal{M}$.

Hopefully, which meaning is intended will be clear from context.

Conditional probability

Let E and F be events and assume $P[F] \neq 0$. The conditional probability of E given F is defined by

$$P[E|F] = P[E \cap F]/P[F].$$

Intuitively, it is the probability that E holds given that F is known to hold.

Example:

Ω	p
1	.2
2	.2
3	.3
4	.1
5	.2

$E = \{1, 2, 3\}$, $F = \{2, 3, 4\}$.

$P[E] = .7$

$P[F] = .6$

$P[E \cap F] = .5$

$P[E|F] = .2/.6 + .3/.6 = 5/6.$

Statistical independence

Formally, events E and F are *statistically independent* if $P[E|F] = P[E]$.

An equivalent definition is that $P[E \cap F] = P[E] \cdot P[F]$.

This is easily seen by dividing both sides by $P[F]$ and applying the definition of $P[E|F]$.

Information-theoretic security

We have been discussing *computational security* – what can Eve learn with a certain success probability in a limited amount of time.

We now turn to *information-theoretic security*, where we remove the limits on Eve and just look at the question from an information-theoretic perspective.
What information is contained in the data at Eve's disposal?

A cryptosystem is information-theoretically secure if $P[m] = P[m|c]$. Thus, c gives no information about m.

This is equivalent to saying that m and c are *statistically independent*.

We also call this *perfect secrecy*.

Base Caesar cipher

The Caesar cipher is said to go back to Roman times.

It encodes the 26 letters of the Roman alphabet $A, B, \ldots, Z$.

Assume the letters are represented as $A = 0$, $B = 1$, $\ldots$, $Z = 25$.

$\mathcal{M} = \mathcal{C} = \mathcal{K} = \{0, \ldots, 25\}$.

$$E_k(m) = (m + k) \bmod 26$$

$$D_k(c) = (c - k) \bmod 26.$$

Full Caesar cipher

Extend base cipher to strings by encrypting each letter separately.

For r-letter strings, we have

$$\mathcal{M}^r = \mathcal{C}^r = \underbrace{\mathcal{M} \times \mathcal{M} \times \ldots \mathcal{M}}_{r},$$

that is, length-r sequences of numbers from $\{0, \ldots, 25\}$. The encryption and decryption functions are

$$E_k^r(m_1 \ldots m_r) = E_k(m_1) \ldots E_k(m_r)$$

$$D_k^r(c_1 \ldots c_r) = D_k(c_1) \ldots D_k(c_r).$$

Note: The key space remains the same; only the message and ciphertext spaces have changed.